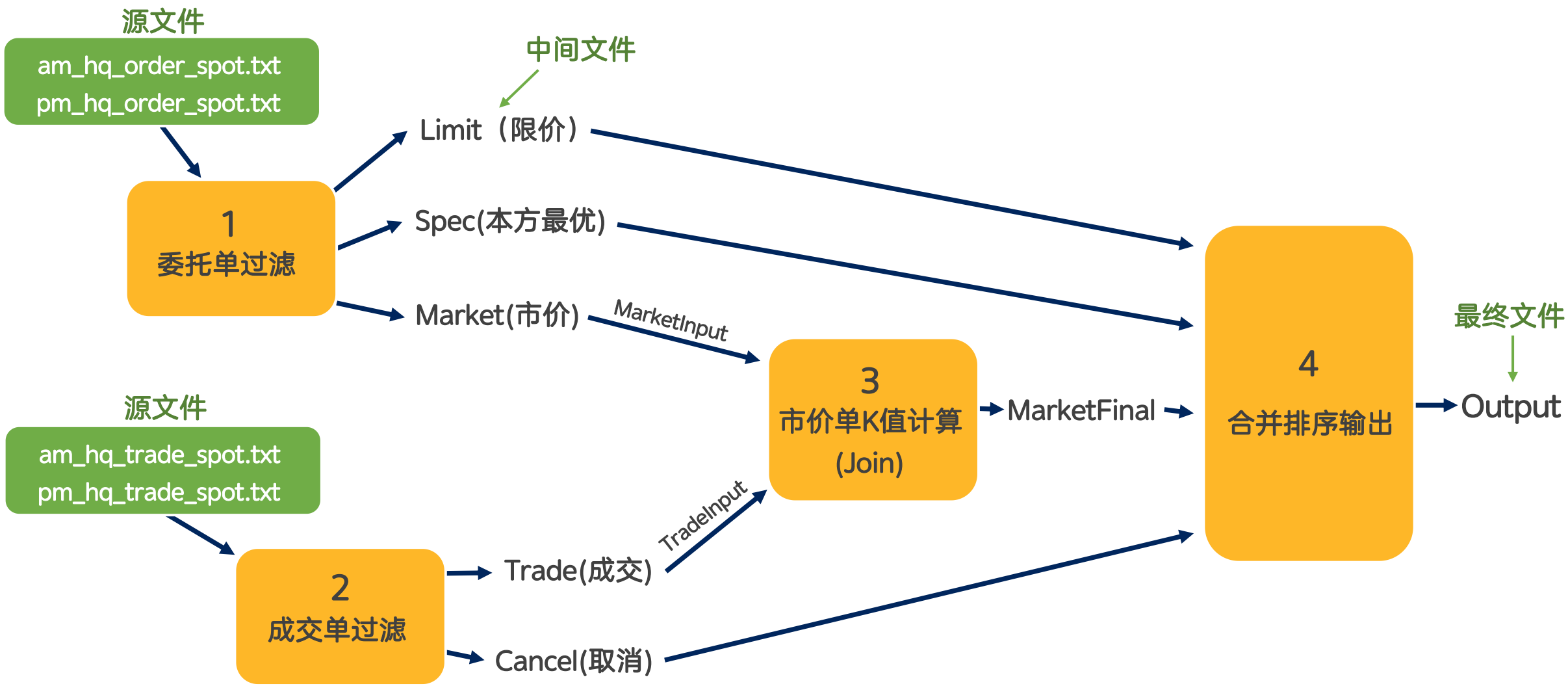
MapReduce Project

12111408郭瑛璞　12112504郭城

目录

CONTENT

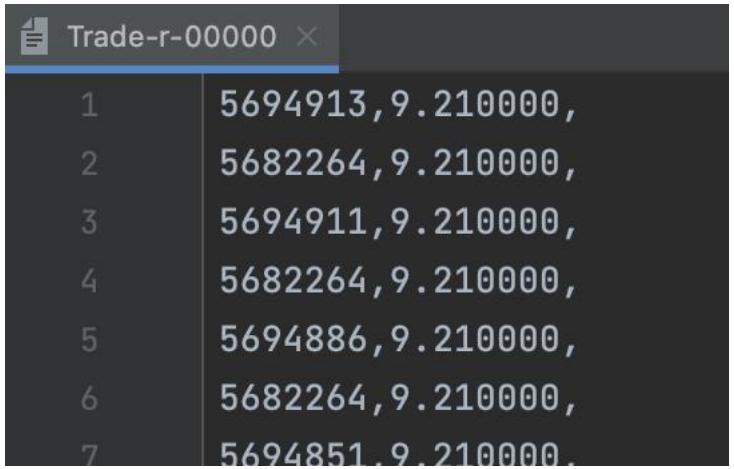
设计思路

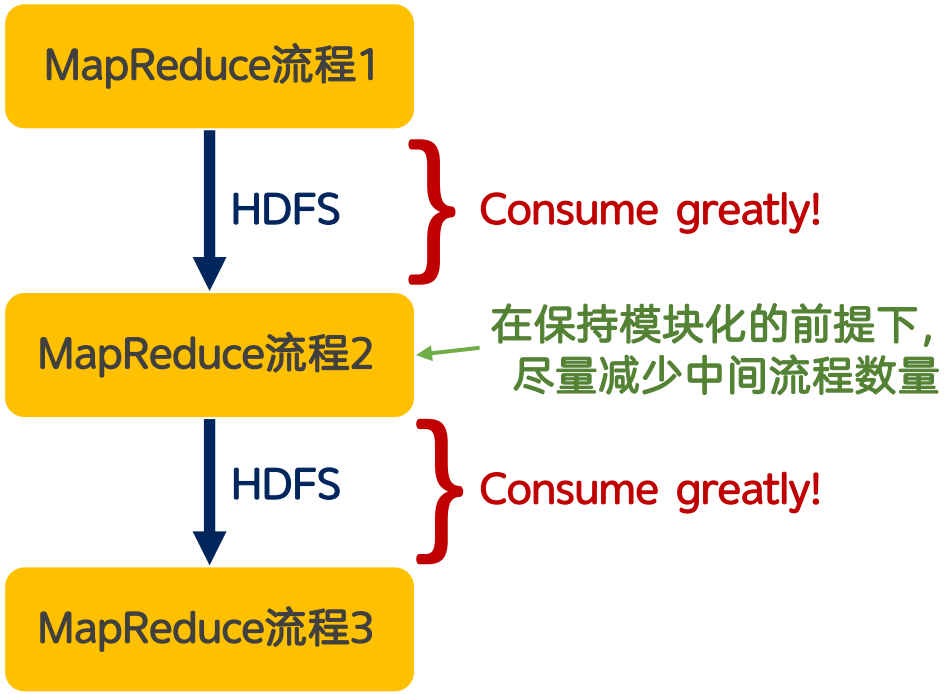

源文件
am_hq_order_spot.txt
pm_hq_order_spot.txt
中间文件
1
委托单过滤
Limit（限价）
Spec(本方最优)
Market(市价)
MarketInput
3
市价单K值计算
(Join)
MarketFinal
4
合并排序输出
最终文件
Output
源文件
am_hq_trade_spot.txt
pm_hq_trade_spot.txt
2
成交单过滤
Trade(成交)
TradeInput
Cancel(取消)

MapReduce优化

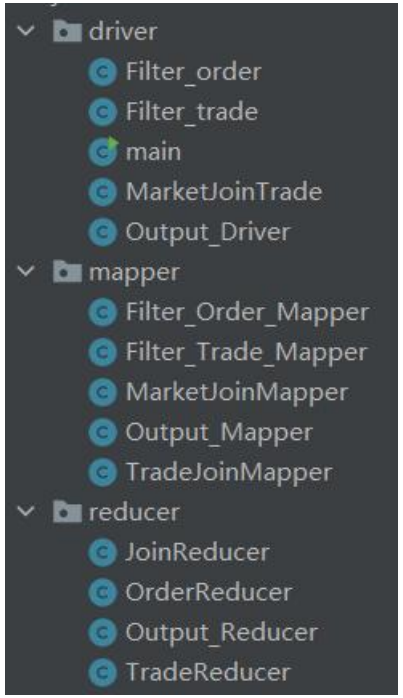
本次project数据共5.05G，我们发现在运行过程中IO的时间占了很大部分。在初步方案中我们将原始数据上午下午分别计算，之后进行一层Map Reduce的合并，增加的这层本地运行时间在2秒以内，但是相比目前方案（上下午使用MultipleInput）增加的文件IO使整个程序在集群上运行时间增加了一分钟，可能原因是集群hdfs需要进行文件分布式处理，读写效率远低于本地，因此在实现过程中需要及时把不需要的字段删除。通过多输入和多输出的方式减少mapreduce总任务的个数，达到加快运行效率的效果。

成交单输出
（卖买方索引，价格）




MapReduce流程1
HDFS
Consume greatly!
MapReduce流程2
在保持模块化的前提下，
尽量减少中间流程数量
HDFS
Consume greatly!
MapReduce流程3

我们的最终程序结构


driver
Filter_order
Filter_trade
main
MarketJoinTrade
Output_Driver
mapper
Filter_Order_Mapper
Filter_Trade_Mapper
MarketJoinMapper
Output_Mapper
TradeJoinMapper
reducer
JoinReducer
OrderReducer
Output_Reducer
TradeReducer

难点分析

K值计算

```
ArrayList<Double> prices = new ArrayList<>();
int K = 0;
boolean t = false;

//key为委托索引 计算市价单K值
for (Text value : values) {
    String s = value.toString();
    char tableId = s.charAt(s.length() - 1);// 提取出辅助字段("o" 或者 "t")
    String fields = s.substring(0, s.length() - 2);// 提取出其他字段

    // 根据辅助字段的值判断属于哪个表
    if (tableId == 't'){//trade成交    委托索引 成交价格 + " t"
        double price = Double.parseDouble(fields.split( regex: ",")[0]);//1?
        if(!prices.contains(price)){
            prices.add(price);
        }
    }

    else {//order委托市价单    委托索引 委托价格 委托数量 委托时间 买卖方向 Order_type + " o
        t = true;//存在order
        order = fields;
    }
}
```

TreeMap是一种数据结构，通常用于实现关联数组或键值对映射。它根据键的自然顺序或根据提供的比较器对键进行排序。在Java中，TreeMap是一个实现了SortedMap接口的类，它使用红黑树实现，确保了键的有序性。

合并排序输出

```
//1.已标记的市价委托单(无撤单)
//2.所有撤单(限价单)
//3.所有限价单

// 20190102093613610,10.300000,1000,2,2,820735,0,2,
// 20190102094711690,0.000000,1800,1,1,1568874,1,2,
Text TIMESTAMP = new Text(values[0]);

context.write(TIMESTAMP,new Text(String.join( delimiter: ",",
        Arrays.copyOfRange(values, from: 1, to: 8))));
```

```
TreeMap<IntWritable,Text> tradeOrder = new TreeMap<>();
TreeMap<IntWritable,Text> cancelOrder = new TreeMap<>();
```

```
//考虑同一时间委托并撤单的情况(共6单)
//即在同一时间有两个orderid，分别是委托单(cancelType = 2)和撤单(cancelType = 1)
for (Text value: values){
    String[] info = value.toString().split( regex: ",");
    IntWritable orderId = new IntWritable(Integer.parseInt(info[4]));
    if (Integer.parseInt(info[6]) == 2) { //委托单存在trade
        tradeOrder.put(orderId, new Text(value));
    } else if (Integer.parseInt(info[6]) == 1) { //撤单
        cancelOrder.put(orderId,new Text(value));
    }
}
```

谢谢

hdfs+MapReduce project report

过滤、K值计算：郭城
排序、report：郭瑛璞